Using Models to Test Process Assumptions within the SEL Recommended Software Development Approach

Paolo Donzelli - Giuseppe Iazeolla

**Laboratory for Computer Science
and CERTIA Research Center**

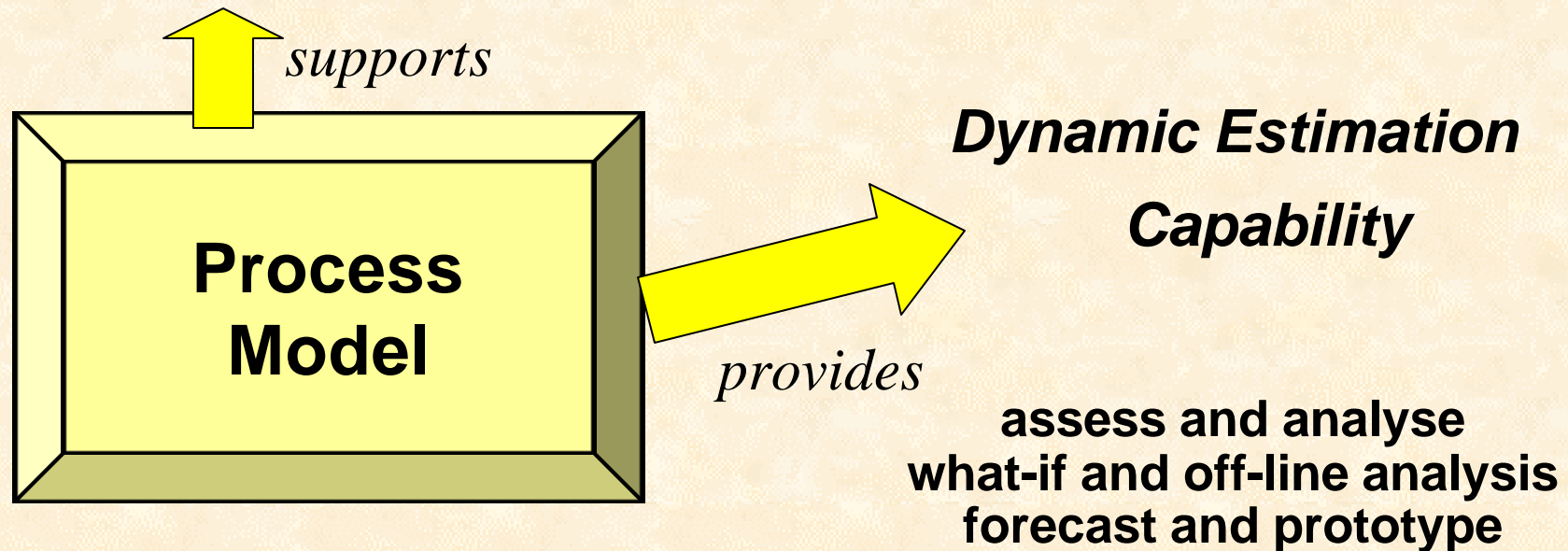**University of Rome "Tor Vergata"
Roma, Italy**

Outline

- **Introduction**

  - Why process modelling, why a hybrid approach, and the suggested approach

- **Building a Process Model**

  - A model of the SEL recommended software development approach

- **Applying the Model**

  - To reproduce some possible software development scenarios

Why Process Modelling

Key objectives of software companies:

- **high quality products**
- **high performance processes**

supports

Process Model

provides

Dynamic Estimation

Capability

assess and analyse
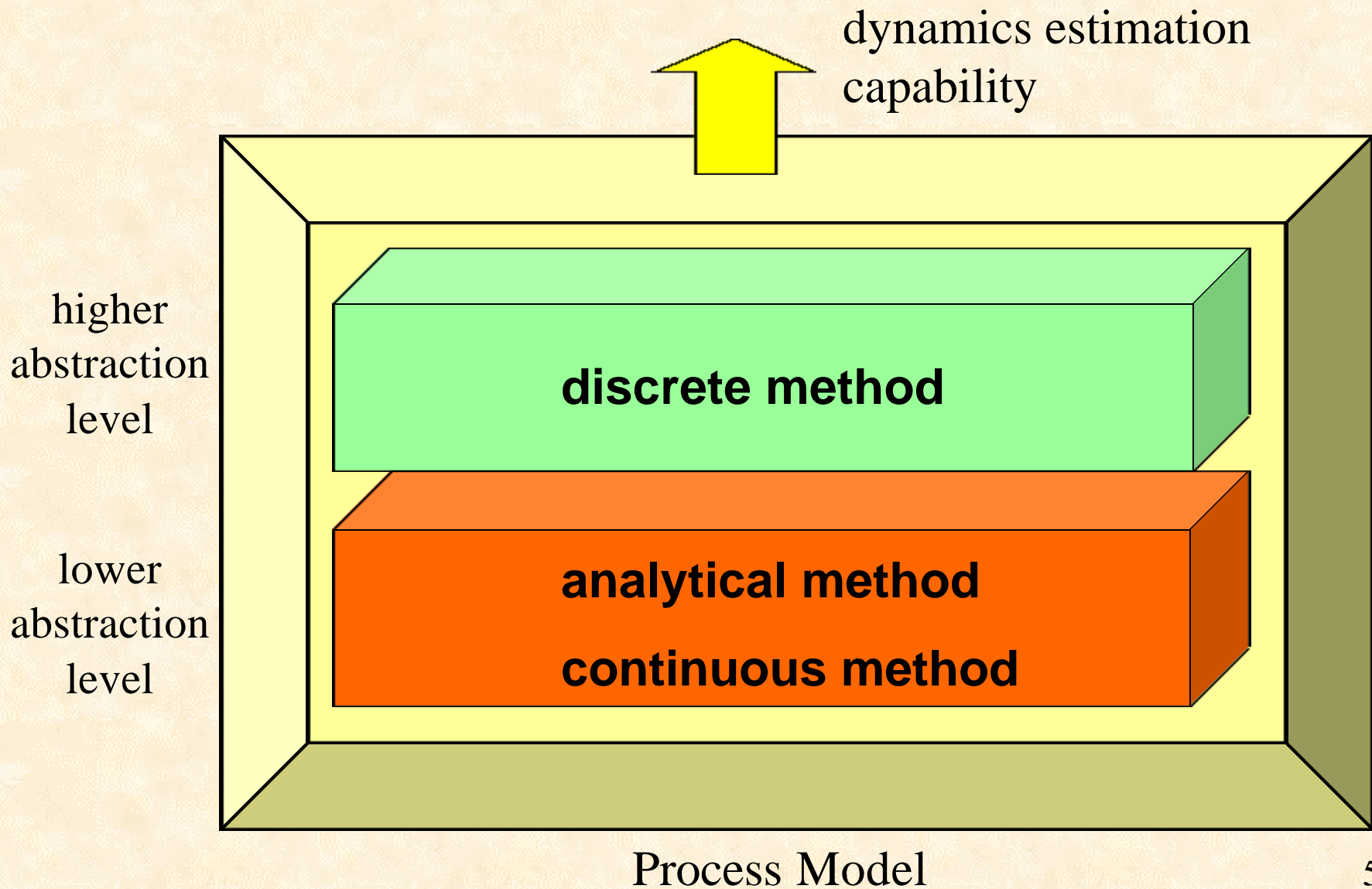**what-if and off-line analysis**
forecast and prototype

Why a Hybrid Approach

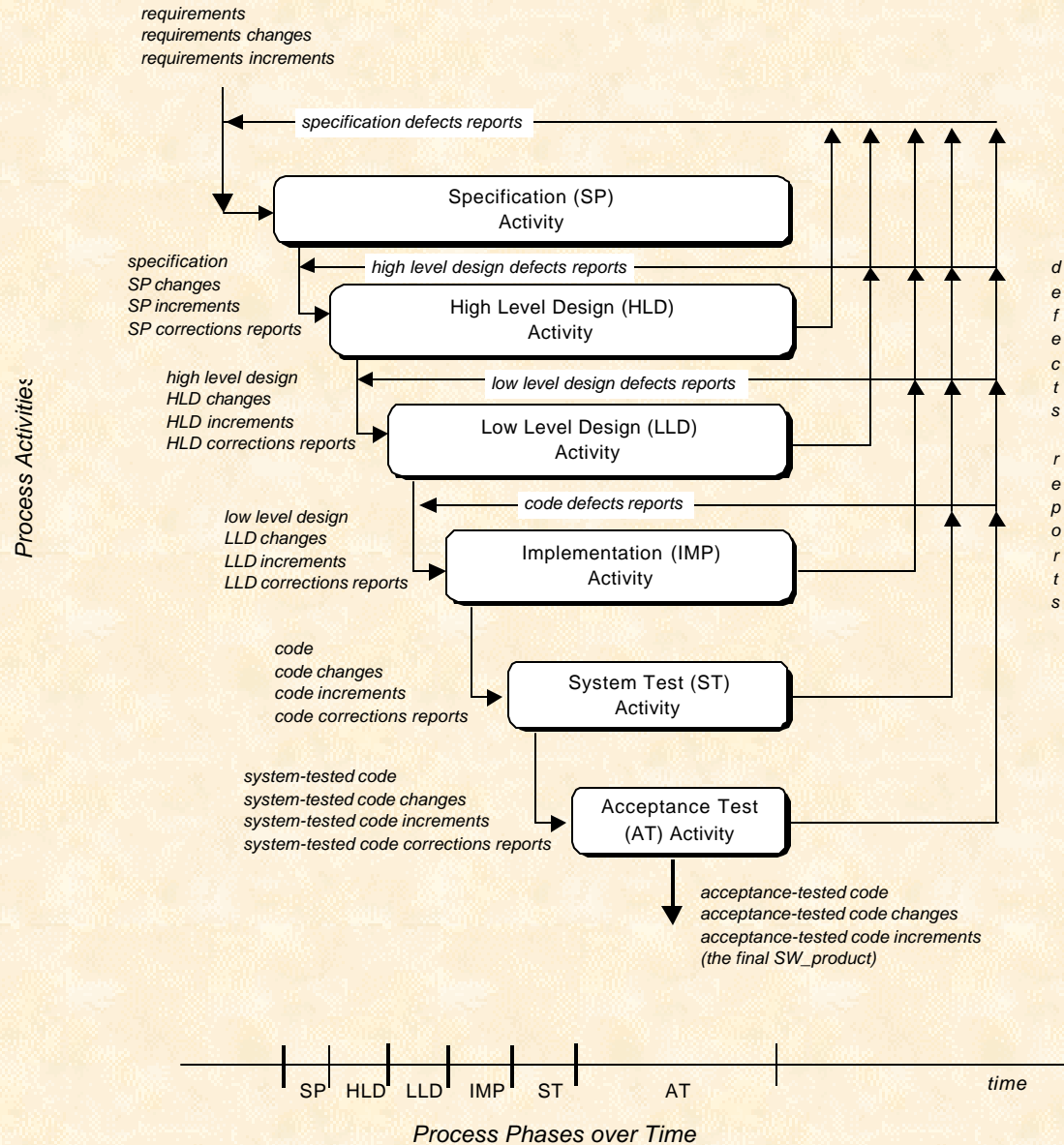**The software process *is composed by various activities*:**

- **some are sequential, others may be performed concurrently**

- **activities exchange artifacts**

- **activities consume resources and may collide**

To model a software process we have to deal with both *discrete system aspects* (start/end of an activity, reception/release of an artifact) and *continuous system aspects* (resource consumption, percentage of developed product).
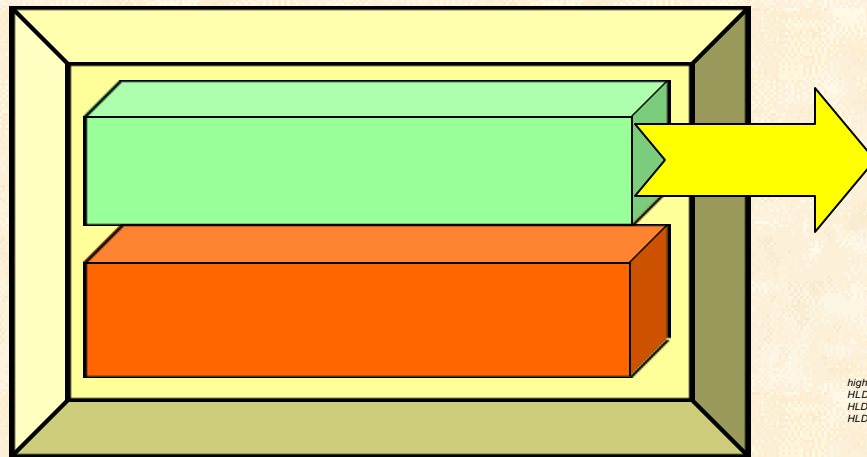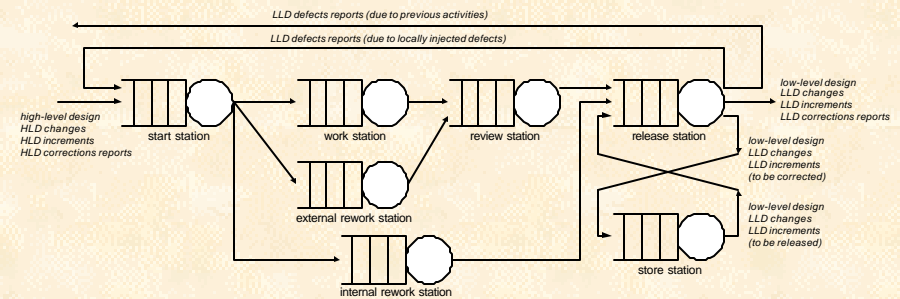
Modelling the SEL Software Process

requirements
requirements changes
requirements increments

Process Activities

specification defects reports

Specification (SP) Activity

specification
SP changes
SP increments
SP corrections reports

high level design defects reports

High Level Design (HLD) Activity

high level design
HLD changes
HLD increments
HLD corrections reports

low level design defects reports

Low Level Design (LLD) Activity

low level design
LLD changes
LLD increments
LLD corrections reports

code defects reports

Implementation (IMP) Activity

code
code changes
code increments
code corrections reports

System Test (ST) Activity

system-tested code
system-tested code changes
system-tested code increments
system-tested code corrections reports

Acceptance Test (AT) Activity

acceptance-tested code
acceptance-tested code changes
acceptance-tested code increments
(the final SW_product)

d e f e c t s   r e p o r t s

SP HLD LLD IMP ST AT *time*

Process Phases over Time
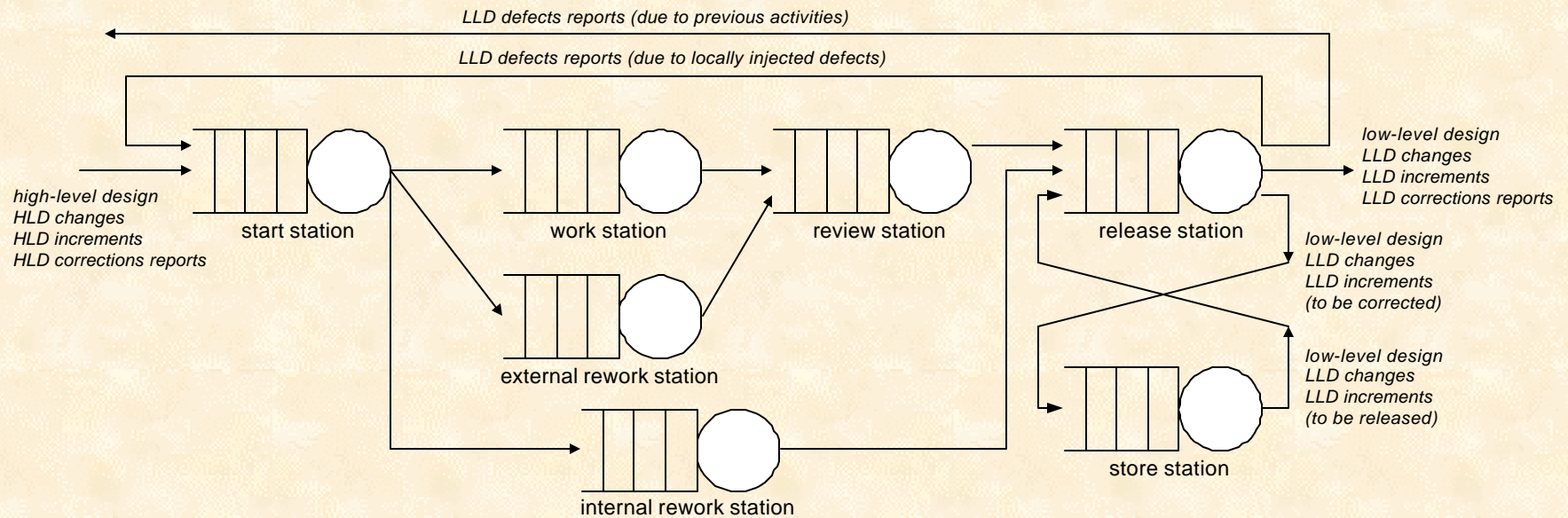
6

The Higher Abstraction Level

**Modelling the
process structure**

The process is modelled by a <u>discrete-event queue net:</u>

- **activities are networked sets of service stations**
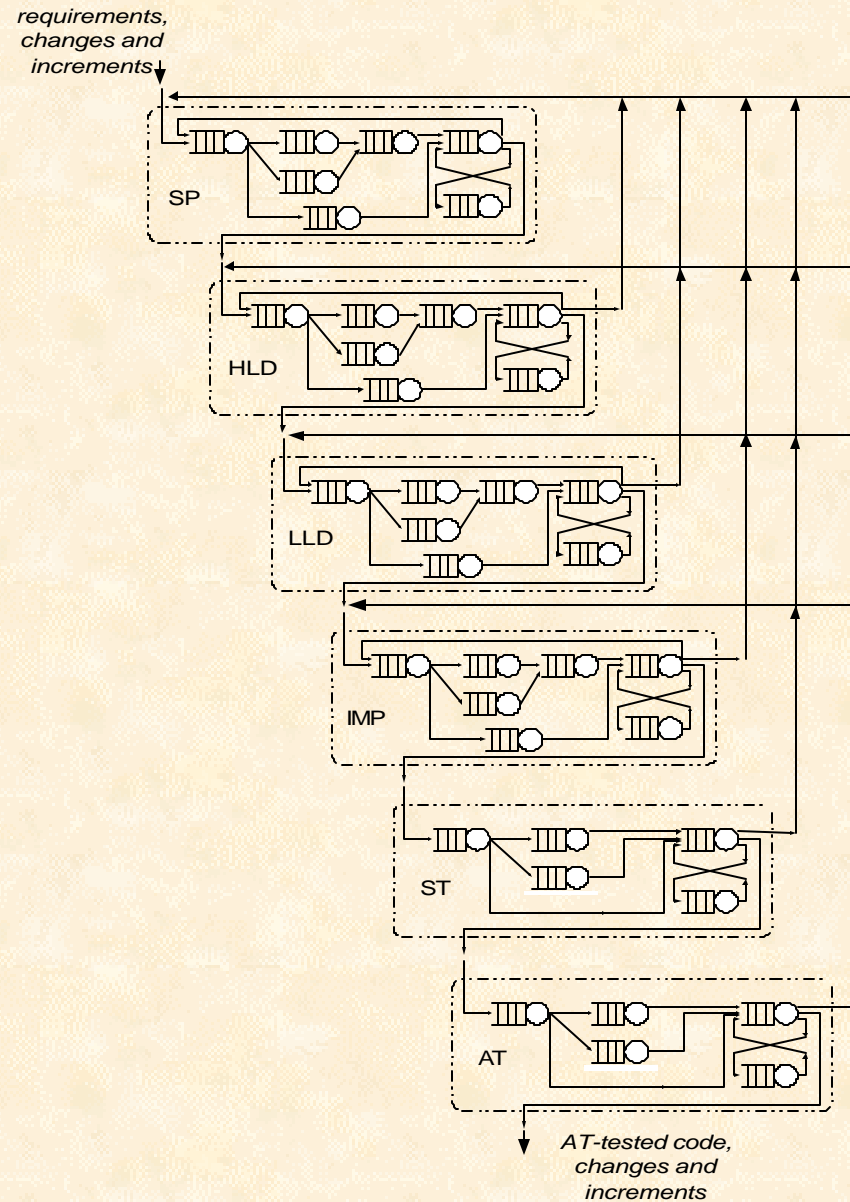
- **artifacts are circulating customers**

7

Higher Abstraction Level

LLD defects reports (due to previous activities)
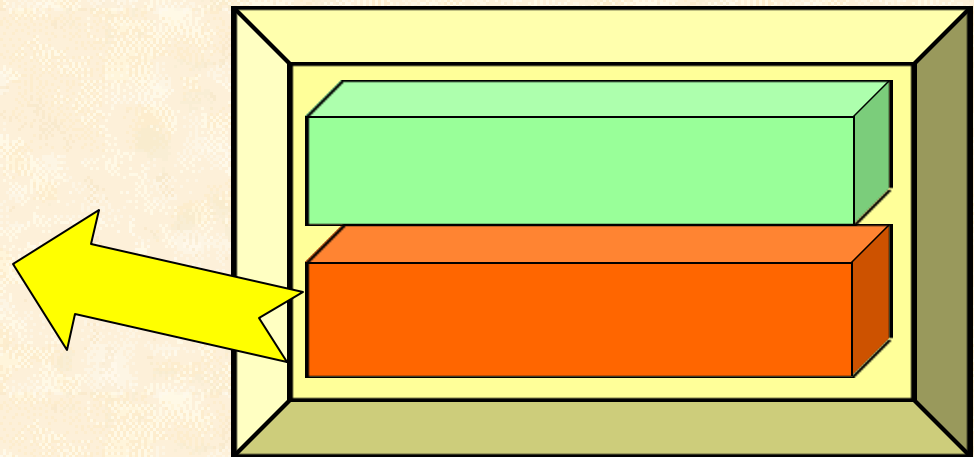
LLD defects reports (due to locally injected defects)

high-level design
HLD changes
HLD increments
HLD corrections reports

start station

work station

review station

release station

external rework station

internal rework station

store station

low-level design
LLD changes
LLD increments
LLD corrections reports

low-level design
LLD changes
LLD increments
(to be corrected)

low-level design
LLD changes
LLD increments
(to be released)

Low-level Design (LLD) Activity

8

requirements,
changes and
increments

SP

HLD

LLD

IMP

ST

AT

AT-tested code,
changes and
increments

The Lower Abstraction Level

Modelling the activities' behaviours

Each activity (service station) is modelled by:

- an analytical average-type function,

- or a continuos type time-varying function,
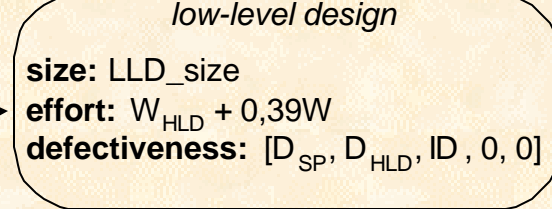
- or a combination thereof.

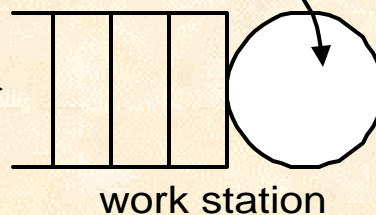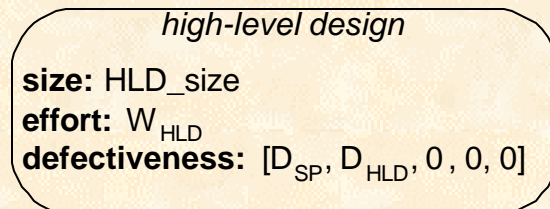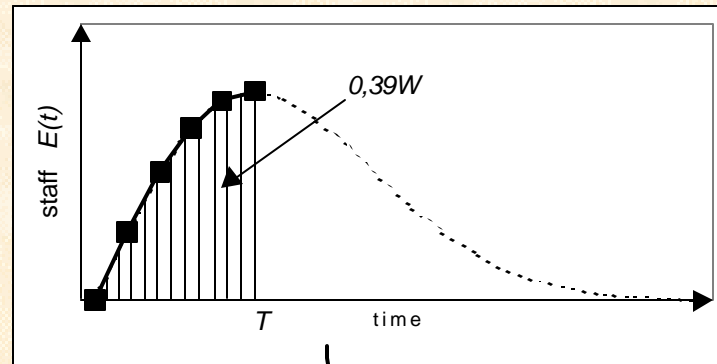# *Lower Abstraction Level of the "Work Station"*

$$LLD\_size = Random(a_1 HLD\_size^{b_1} + c_1)$$

$$T = a_2 LLD\_size^{b_2} + c_2$$

$$W = a_3 LLD\_size^{b3} + c_3$$
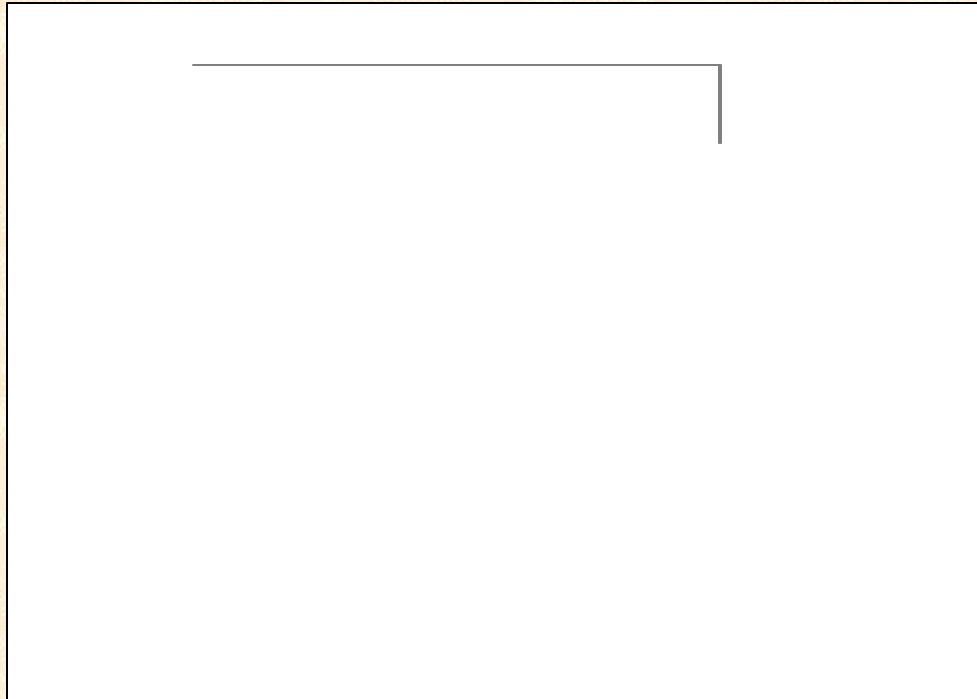
$$E(t) = W \frac{t}{T^2} e^{-\frac{t^2}{2T^2}}$$

high-level design

size: HLD_size
effort: W_{HLD}
defectiveness: $[D_{SP}, D_{HLD}, 0, 0, 0]$

work station

low-level design

size: LLD_size
effort: $W_{HLD} + 0,39W$
defectiveness: $[D_{SP}, D_{HLD}, ID, 0, 0]$

11

Applying the Model

Two possible software development scenarios are simulated:

- with **<u>stable set of requirements</u>** (1500 FPs)

- with a certain amount of **<u>requirements instability</u>**

The main process attributes are *effort* (W), *delivery time* (T), *productivity* (P), *rework percentage* (RWK), and *product defect density* (DFD).
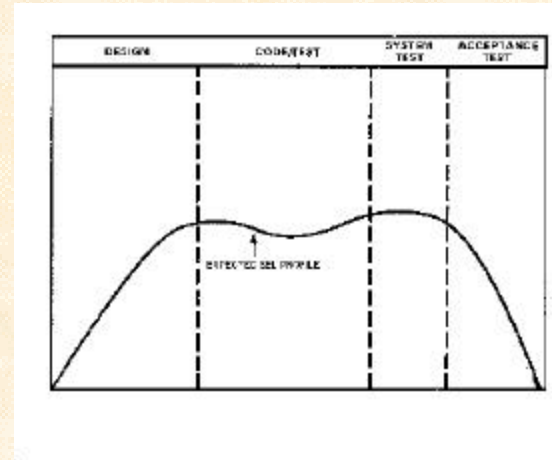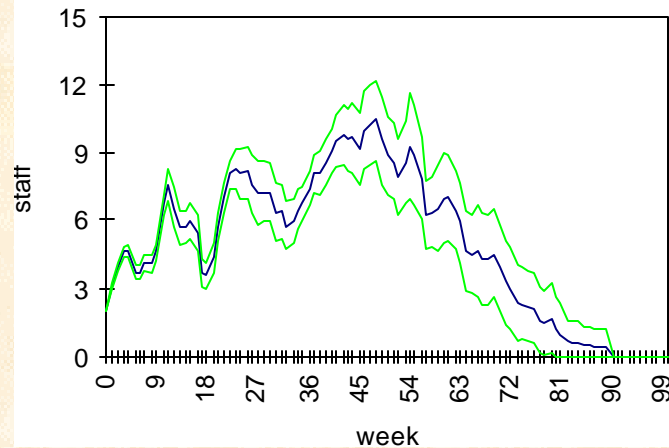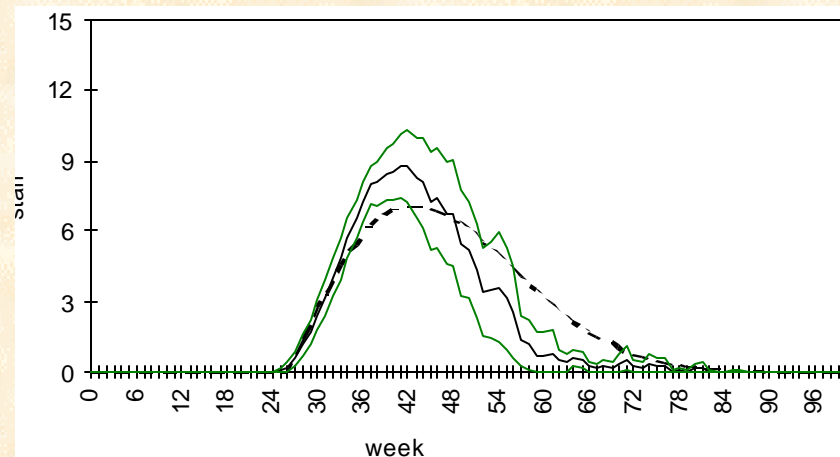
Stable Requirements – Simulation Results

Stable Requirements – Comparison with SEL

Attribute	Model	Conf. 95%	SEL
Final Size	116 KLOC	+/- 20	116
Effort	500 PW	+/- 60	600
Delivery	78 W	+/- 3	63
Productivity	5.8 LOC/p-hour	+/- 1.8	5.3
Rework %	17 %	+/- 3%	/
Defect Density	0.9 Defects/KLOC	+/- 0.3	/
Average Staff	6.5 P	+/- 1	9.5

Stable Requirements – Comparison with SEL

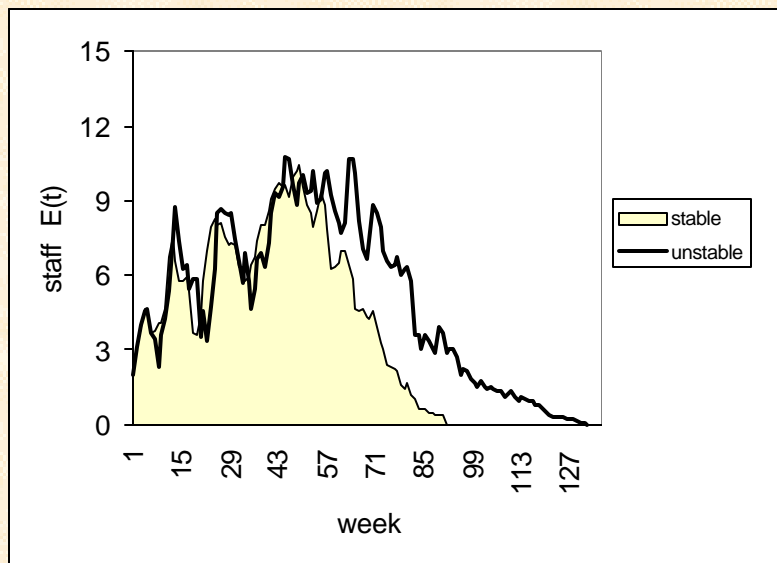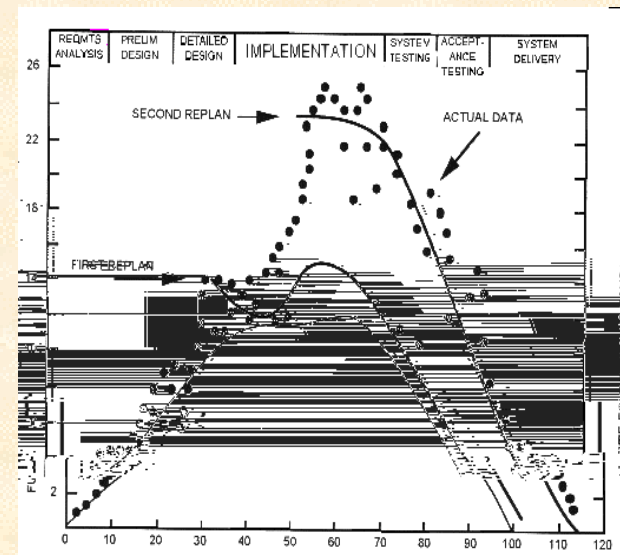**project staff profile**

activity staff profile

15

Effects of Instability on the Staffing Profile

Simulation Results

A real project

16

Conclusions

Simulation results demonstrate the capability of the described model of

- **reproducing empirically-known facts**

- **being adopted as tool to test process assumptions**

The suggested approach allows high model flexibility and reusability:

- **easy extension to other process paradigms and easy hierarchical modelling of activities's details;**

- **adaptable to the maturity of the target environment, and updatable to follow its evolution (CMM, QIP).**